Arbogast – Origine d'un outil de dérivation automatique

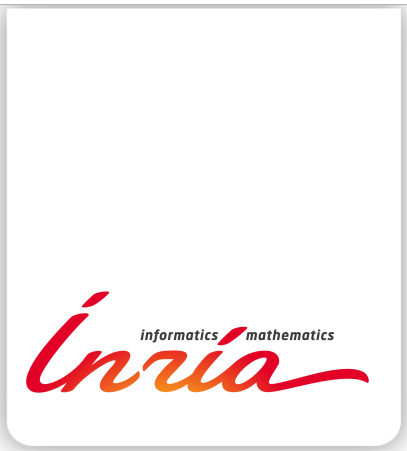
Isabelle Charpentier, Jean-Pierre Friedelmeyer, Jens Gustedt

Arbogast – Origine d'un outil de dérivation automatique

Isabelle Charpentier **Jean-Pierre Friedelmeyer** **Jens Gustedt**

RESEARCH REPORT
N° 8911
Mai 2016
Project-Team Camus

ISSN 0249-6399 ISRN INRIA/RR--8911--FR+ENG



Arbogast – Origine d'un outil de dérivation automatique

Isabelle Charpentier [*] Jean-Pierre Friedelmeyer [† ‡]

Jens Gustedt [§]

Équipe-Projet Camus

Rapport de recherche n° 8911 — Mai 2016 — 8 pages

Résumé : Les dérivées que considère Arbogast « sont moins des dérivées de quantités que des dérivées d'opérations ». Cette assertion extraite « du calcul des dérivations » est confirmée tout au long de ce traité. Polynômes de Taylor, différentielles divisées, formules de récurrence, origine des dérivations introduits par Arbogast, présents aujourd'hui dans tous les logiciels de différentiation de haut degré, révèlent clairement l'origine de la discipline dite de différentiation automatique.
Cet article rend hommage à Arbogast en rappelant les liens fondamentaux entre « calcul des dérivations » et différentiation automatique, et en lui dédiant un outil de dérivation/différentiation automatique fonctionnelle.

Mots-clés : Analyse algébrique ; Règle de dérivation ; Différentiation de haut degré ; Modular C

[*] ICube, CNRS and Université de Strasbourg, France
[†] Archives Henri-Poincaré, CNRS et Université de Lorraine, France.
[‡] IREM de Strasbourg, France.
[§] Inria, France

RESEARCH CENTRE
NANCY – GRAND EST

615 rue du Jardin Botanique
CS20101
54603 Villers-lès-Nancy Cedex

`Arbogast` – Origin of an automatic derivation software

Abstract: The derivatives considered by Arbogast "*are less derivatives of quantities than derivatives of operations*". This assertion drawn from "*du calcul des dérivations*" is confirmed all along this treatise. Taylor polynomials, Taylor coefficients, recurrence formulas, *origin of derivations* introduced by Arbogast, currently present in all the higher-order differentiation software, clearly reveal the origin of the discipline called automatic differentiation.

This article honours Arbogast by recalling the fundamental links between « calcul des dérivations » and automatic differentiation, and by dedicating an functional automatic derivation/-differentiation software to his name.

Key-words: Algebraic analysis ; Chain rule ; higher-order differentiation ; Modular C

Abridged English version

At Strasbourg in 1800, L.F.A. Arbogast (1759–1803) writes "*du calcul des dérivations*" stating the principle of deriving the compound function $\varphi \circ \psi$, the second of which is the power series (1). As noted in the preface, the derivatives he considers "*are less derivatives of quantities than derivatives of operations*". Arbogast's treatise is immediately spotted by English mathematicians such as Woodhouse, De Morgan, Babbage and Herschel or Cayley. "*Du Calcul des dérivations*" is obviously at the origin of the scientific discipline called automatic diffentiation (AD) [12]. In a nutshell, this considers any computer code as a sequence of statements comprising operators, elemental functions and variables. The differentiation performed at the level of such operators and functions is propagated to statements thanks to the chain rule at order 1, or the generalized chain rule at higher-orders. This is present in Arbogast's treatise, see (3). Nevertheless, by ignorance, despite controversy [15, 10], the generalized chain rule is known as Faà di Bruno's formula (4).

This article has several goals, the most important of which are to revive the memory of Arbogast's work in France and to dedicate a functional automatic derivation/differentiation software to his name. Section 2 recalls definitions to provide access to Arbogast's rule (3). These include the power series (1), the factorial, the scaled derivatives (2) known as Taylor coefficients in AD, and the rule (3) together with its so-called partitions. Faà di Bruno's formula (4), written using more affordable notations, is shown for comparison. The generalized chain rule has little interest in practice. As demonstrated in Section 3, the writting of ψ as a power series is in straigthforward agreement with the Taylor polynomials (5). AD is applied to Taylor polynomial variables at the level of the operators and the elemental functions of the computer code. Higher-order AD is implemented from efficient recurrence formulas [13, pp. 305–308] such as the Leibniz' formula written in terms of Taylor coefficients (6) for the sake of numerical stability. This formula as many others, see (7) for the exponential, are sketched in Arbogast's treatise. Computed derivatives are "exact" up to the machine precision, except when the numerical code implements some fixed point algorithm, the number of iterations of which is insufficient to guarantee the differentiation results [9]. Mathematical functions studied by Arbogast's contemporaries (Bessel, Gauss, Kramp) thus require a particular differentiation (8)-(10).

Section 4 presents the tool box `arbogast` for AD. It builds upon an extension of the C programming language, *Modular C* [14], and offers the ability to contextualize C functions. In the present case, any "contextualized" function may be either compiled as a classical numerical function, or as a higher order derivative AD one. In the general case, the only changes to make eligible a function for AD are (1) that variables must be typed by a symbolic type name to be interpreted as Taylor polynomials and (2) that related arithmetic expressions must be quoted by special "context" markers. `Arbogast` does all the necessary rewriting of types, functions and operators during compilation, such that no dynamic dependencies remain in the executable. This process is illustrated considering Héron's method for the approximation of $\sqrt[kappa]{x}$. The C code implementing this method for a double variable x presented in Code 1 may be compiled in C and *Modular C*. As shown in Code 2, *Modular C* introduces generality through the generic type `RC`, corresponding to any of the six float types. The six related functions are generated at compile time. Note that *Modular C* allows for the use of special characters in the code. The contextualization may be observed in Code 3. Execution times of *arbogast* codes can be substantially faster than comparable AD strategies for other programming languages.

1 Introduction

Strasbourg, An VIII (1800), L.F.A. Arbogast (1759–1803) préface le « calcul des dérivations » comme suit. « On s'est proposé dans cet ouvrage d'offrir un genre de calcul qui embrasse la théorie des suites, et dont le calcul différentiel n'est qu'un cas particulier : il fournit des procédés qui abrègent des opérations laborieuses, et des formules qui facilitent les recherches dans des matières compliquées [...] » Quelques lignes plus bas, il précise encore « les dérivées que je considère sont moins des dérivées de quantités que des dérivées d'opérations, comme l'Algèbre est moins un calcul de quantités que d'opérations arithmétiques ou géométriques à exécuter sur les quantités. » Les travaux d'Arbogast furent immédiatement remarqués Outre-Manche, leur nature calculatoire séduisant des mathématiciens anglais du 19$^{\text{ème}}$ siècle tels que Woodhouse, Babbage et Herschel, De Morgan ou Cayley.

Le calcul des dérivations d'Arbogast est clairement à l'origine de la discipline mathématico-informatique nommée différentiation automatique (DA) [12]. En quelques mots, celle-ci considère chaque code de calcul comme une séquence d'instructions, constituées d'opérateurs, de fonctions élémentaires et de variables. Deux des arguments théoriques principaux sont la règle de dérivation des fonctions composées et la généralisation de cette règle au cas des dérivées d'ordre supérieur qui, appliquée au niveau des opérations et fonctions élémentaires, permettent de dériver des instructions complexes et, de proche en proche, le code complet. Pourtant, par méconnaissance historique, malgré la controverse [15, 10], la formule de dérivation d'ordre élevé porte le nom de Faà di Bruno [11].

Cette Note a plusieurs objectifs. Il s'agit tout d'abord de « faire revivre en France le calcul symbolique

d'Arbogast et de Servois, calcul qui condense et mémorise une foule de théorèmes, [...] » comme souhaité en 1855 dans les « Nouvelles Annales de Mathématiques » [18, 17] lors de la revue d'un traité de Carmickaël, et de le faire en français. Dans la Section 2, une brève étude du traité d'Arbogast est conduite pour présenter vocabulaire et définitions permettant d'accéder à sa règle de dérivation qui, par nature, est différente de la formule de Faà di Bruno. Comme nous le verrons en Section 3, la DA n'utilise ni l'une ni l'autre puisqu'elle implante des formules de récurrence spécifiques aux opérateurs et fonctions élémentaires, formules déjà présentes dans [2]. La Section 4 propose une implantation innovante de la DA, conçue sur les Terres d'Arbogast, confirmant le caractère intemporel de ses travaux sur la dérivation des fonctions de polynômes. Conclusions et une courte biographie de d'Arbogast sont proposées en Section 5.

2 Elements « du calcul des dérivations »

Les travaux d'Arbogast concernent la dérivation de haut degré de fonctions composées $\varphi \circ \psi(x)$ lorsque ψ est écrite en série entière,

$$\psi(x) = \psi_0(x) = \sum_m \psi_m x^m, \tag{1}$$

les ψ_m (notés α, β, γ, $\cdots$ dans [2]) désignant à la fois les coefficients associés aux puissances x^m, et les séries entières $\psi_m(x)$ issues de chacun des ψ_m, sans hypothèse de convergence. Sous l'hypothèse anachronique d'une fonction ψ de classe C^∞, les ψ_m sont les différentielles divisées $\underset{C}{\mathrm{D}}{}^m\psi$ (n° 39, p. 33), aujourd'hui notées

$$\psi_m = \underset{C}{\mathrm{D}}{}^m\psi = \frac{1}{1.2.3.\cdots.m}\frac{\partial^m\psi}{\partial x^m}(x) = \frac{1}{m!}\frac{\partial^m\psi}{\partial x^m}(x) = \frac{1}{m!}\psi^{(m)}(x). \tag{2}$$

Arbogast nomme « factorielle, les produits de facteurs dont les premières différences sont constantes » (n°420, p. 364), la notation $m!$ étant introduite par Kramp en 1808 [16].

La règle d'Arbogast pour la dérivation de haut degré (n°41, p. 34) est récursive,

$$\underset{C}{\mathrm{D}}{}^m.\varphi\psi_0 = \underset{C}{\mathrm{D}}{}^m\varphi\psi_0.\psi_1^m + \underset{C}{\mathrm{D}}{}^{m-1}\varphi\psi_0.\mathrm{D}.\psi_1^{m-1} + \underset{C}{\mathrm{D}}{}^{m-2}\varphi\psi_0.\underset{C}{\mathrm{D}}{}^2.\psi_1^{m-2} + \ldots + \underset{C}{\mathrm{D}}\varphi\psi_0.\underset{C}{\mathrm{D}}{}^{m-1}.\psi_1, \tag{3}$$

distinguant, pour $1 \le k \le m$, les différentielles divisées $\underset{C}{\mathrm{D}}{}^k\varphi\psi_0$ de φ évaluées en ψ_0, des dérivées divisées $\underset{C}{\mathrm{D}}{}^k.\varphi\psi_0$ de la fonction composée $\varphi \circ \psi$ évaluées en 0 (n°10, p. 7). Arbogast appelle $\varphi\psi_0$ « origine des dérivations ». La justesse de cette dénomination est illustrée dans la Section 3. La remarque n°52 (p. 43) décrit en détail le développement des termes ψ_1^m, $D.\psi_1^{m-1}$, $\underset{C}{\mathrm{D}}{}^2.\psi_1^{m-2}$, ..., $\underset{C}{\mathrm{D}}{}^{m-1}.\psi_1$ et les partitions d'entiers impliquées. Par exemple, l'expression $\underset{C}{\mathrm{D}}{}^n.\psi_1^p$ désigne le coëfficient du $n+1^e$ terme, dans le développement de la puissance p du polynôme $\psi_1 + \psi_2 x + \psi_3 x^3 + \cdots$. Ce coefficient est formé de la somme de tous les produits $\psi_1^{i_0}.\psi_2^{i_1}.\psi_3^{i_2}.\psi_3^{i_4}.\cdots$ que permet la partition de n en p entiers positifs ou nuls i_0 ,i_1, i_2, $i_3, \cdots$ telles que $i_0 + i_1 + i_2 + i_3 \cdots = p$ et $i_1 + 2i_2 + 3i_3 + \cdots = n$.

Usant de notations plus abordables, devenues classiques, Faà di Bruno propose en 1857 « une formule propre à calculer immédiatement la dérivée d'un ordre quelconque d'une fonction de fonction » [11],

$$D_x^m\varphi = \sum \frac{\prod(m)}{\prod(i)\prod(j)\cdots\prod(k)} D_y^p\varphi.\left(\frac{\psi'}{1}\right)\left(\frac{\psi''}{1.2}\right)\left(\frac{\psi'''}{1.2.3}\right)\cdots\left(\frac{\psi^{(l)}}{\prod(l)}\right), \tag{4}$$

pour la dérivation de la fonction composée $v(x) = \varphi(y) = \varphi \circ \psi(x)$ par rapport à x. La somme porte sur les partitions précédentes. L'histoire de la formule de Faà di Bruno et de sa démonstration tardive est controversée [15, 10]. La formule s'écrit aujourd'hui à l'aide des polynômes de Bell [3] comprenant la somme sur les partitions.

Considérées similaires, la règle d'Arbogast et la formule de Faà di Bruno sont néanmoins conceptuellement différentes. L'écriture de ψ en série entière affirme le caractère calculatoire de la première et sied parfaitement à la différentiation automatique.

3 Différentiation automatique de haut degré

Dans un code informatique, la différentiation automatique (DA) est effectuée en distinguant les variables indépendantes x et y, des variables dépendantes de celles-ci par une relation fonctionnelle $z = \psi(x, y)$ ou $w = \varphi(z) = \varphi(\psi(x, y))$, par exemple. La dérivation est propagée opérateur par opérateur, fonction par fonction, les variables dépendantes intervenant à leur tour dans la dérivation. Les règles classiques sont appliquées au niveau de chaque opérateur et de chaque fonction (ln, exp, cos, par exemple). Ici réside une différence mineure entre Arbogast et DA car une fonction comme $\exp(-x^2)$ est dérivée directement chez le premier, tandis que la seconde considère successivement $z = x^2$, $w = -z$, puis $\exp(w)$.

Fondamentaux – On suppose désormais que les variables indépendantes dépendent implicitement d'une même variable t quelconque.

Pour des raisons de stabilité numérique [13], la DA de haut degré est implantée à l'aide des différentielles divisées $x_m = \frac{1}{m!}\frac{\partial^m x}{\partial t^m}$ de x, appelées coefficients de Taylor. Développée en 0, le polynôme de Taylor d'ordre M de la fonction $x(t)$ vérifie

$$x(t) \simeq x_0 + tx_1 + t^2x_2 + \cdots + t^M x_M = \sum_{m=0}^{M} t^m x_m. \tag{5}$$

La DA de haut degré repose sur des relations de récurrence efficaces dont le nombre d'opérations est quadratique en M [13, pp. 305–308], telle la formule dite de Leibniz

$$z_m = \sum_{j=0}^{m} x_j y_{m-j}, \tag{6}$$

pour la différentiation du produit $z(t) = x(t)y(t)$ par rapport à t. Comme noté par Arbogast (n°86, p. 70), l'usage des différentielles divisées permet d'occulter les coefficients binomiaux présents dans la formule originale. De même, la formule de récurrence de dérivation de la fonction exponentielle $z(t) = \exp(x(t))$,

$$\tilde{z}_m = \sum_{j=1}^{m} z_{m-j}\tilde{x}_j, \quad \text{avec } \tilde{z}_m = mz_m \text{ et } \tilde{x}_j = jx_j, \tag{7}$$

figure chez Arbogast au n°15 (p. 11). Plus généralement, les formules de récurrence de fonctions élémentaires univariées vérifiant une équation différentielle ordinaire sont construites à partir de celle-ci [13], comme le propose Arbogast en d'autres mots. La fonction $\varphi(x(t))$ soumise à dérivation n'est évaluée qu'une fois, en $x_0 = x(0)$, pour calculer l'origine des dérivations $z_0 = \varphi(x_0)$ qui initie la récurrence.

Les dérivées calculées numériquement par DA sont « exactes » à la précision machine près. Néanmoins, une perte de précision numérique peut survenir lorsque les codes de simulation comprennent une méthode de point fixe [9], comme dans l'implantation des fonctions spéciales [1] proposées par les contemporains d'Arbogast (Bessel, Gauss, Kramp, ...). Les solveurs linéaires ou les solveurs non-linéaires utilisés pour la résolution de problèmes d'équations aux dérivées partielles nécessitent également une DA adaptée. Ces deux exemples sont rapportés ci-dessous.

Fonctions spéciales – La fonction de Kramp, aujourd'hui appelée fonction de Faddeeva, peut être implantée numériquement par le biais d'un dévelopement en série dont le nombre de termes, calculé *a priori*, suffit à l'évaluation précise de la fonction, mais peut être insuffisant pour garantir les résultats d'une DA appliquée directement au code de calcul [9]. La DA des fonctions spéciales [1] satisfaisant l'équation différentielle ordinaire de second ordre écrite sous la forme générale

$$\alpha(x)\varphi^{(2)}(x) + \beta(x)\varphi^{(1)}(x) + \gamma(x)\varphi(x) = 0, \tag{8}$$

est implantée avec un nombre d'opérations en $O(M^2)$ [6] à partir de l'origine des dérivations (9),

$$v^{(0)} = \varphi^{(0)} = \varphi(x_0), \quad v^{(1)} = \varphi^{(1)}x^{(1)}, \tag{9}$$

$$v^{(2)} = \frac{-\gamma v^{(0)}(x^{(1)})^3 - \beta v^{(1)}(x^{(1)})^2 + \alpha v^{(1)}x^{(2)}}{\alpha x^{(1)}}, \quad \text{pour } \alpha x^{(1)} \neq 0, \tag{10}$$

en appliquant les formules de DA classiques aux opérateurs de (10). Les cas particuliers, dont $x^{(1)} = 0$, sont abordés dans [7].

Solveurs non-linéaires – Contemporain d'Arbogast, Brisson [4] utilise des développements en séries d'opérateurs pour résoudre des problèmes d'équations aux dérivées partielles linéaires. Ultérieurement, voir [5] et les références qu'il comprend, plusieurs auteurs ont proposés de résoudre numériquement des problèmes non-linéaires suffisament réguliers, comprenant des équations différentielles ordinaires ou algébriques, des équations aux dérivées partielles, qu'ils soient posés sous forme de problèmes aux limites, de problèmes aux valeurs propres, ou de problèmes paramétriques. Toutes ces méthodes reposent sur un même argument, introduire des polynômes de Taylor dans le problème non-linéaire pour en déduire une séquence récurrente de problèmes linéaires.

Ces problèmes peuvent être résolus en considérant l'équation résiduelle écrite sous la forme générale

$$\varphi(x(t)) = 0, \tag{11}$$

où φ est une fonction supposée analytique. Le vecteur d'état $x(t)$ est approché par le polynôme (5). Introduire (5) dans (11), puis utiliser l'analycité de φ suivant les puissances de t permettent d'isoler le coefficient de Taylor x_m en utilisant la règle d'Arbogast (3)

$$0 = (\varphi \circ x)_m = \varphi_1 x_m + \{\varphi_{m|x_m=0}\}, \tag{12}$$

où φ_1 est le Jacobien de φ et le terme de haut degré $\{\varphi_{m|x_m=0}\}$ dépend des coefficients de Taylor $x_0, \cdots, x_{m-1}$. On en déduit la séquence de systèmes linéaires (13) issue de la solution du problème linéarisé $\varphi_1 x_0 = \varphi(x_0)$,

$$\varphi_1 x_m = -\{\varphi_{m|x_m=0}\}, \quad \text{pour } m = 1, \cdots, M, \tag{13}$$

permettant de déterminer, un après l'autre, les coefficients de Taylor x_m. Jacobien et seconds membres de haut degré $\{\varphi_{m|x_m=0}\}$ sont calculables par DA de manière simple et efficace. En effet, comme les coefficients $x_0, \cdots, x_{m-1}$ sont connus à l'ordre m, le choix de $x_m = 0$ permet d'evaluer φ_m. Le résidu à l'ordre M est utilisé pour évaluer le domaine de validité de l'approximation (5) *a posteriori*.

4 Implantation fonctionnelle d'`arbogast`

Les liens entre travaux d'Arbogast et DA vont bien au-delà de la formule de dérivation (4), celle-ci ne présentant que peu d'intérêt d'un point de vue numérique. Désormais φ est un code de calcul.

Implantation classique – La mise en œuvre des fondamentaux (Section 3) est devenue classique et de nombreux logiciels de DA de haut degré sont référencés sur le site communautaire `autodiff.org`. Ecrits pour divers langages de programmation, ces logiciels partage les caractéristiques suivantes : *(i)* généralité, c'est-à-dire capacité à dériver les instructions d'un code de calcul sans autre information que les variables indépendantes et l'ordre de troncature ; *(ii)* variables représentant des polynômes de Taylor ; *(iii)* formules récurrentes telles (6) et (7).

Pour les langages de programmation qui le permettent, les logiciels actuels reposent sur la technique de surcharge d'opérateurs et de fonctions pour attacher les calculs de dérivées aux opérateurs et fonctions élémentaires du langage de programmation. Il s'agit de définir une bibliothèque comprenant un ou plusieurs types « polynômes de Taylor » `T` et les méthodes codant les formules de récurrence pour toute opération ou fonction élémentaire impliquant une variable de type `T`. L'utilisateur choisit les variables indépendantes en les déclarant de ce type et la dérivation est effectuée à l'exécution du code φ, ce qui peut nécessiter de déclarer de type `T` toutes les variables dépendantes comme dans la plupart des outils de DA.

Implantation fonctionnelle – `Arbogast` est un outil de DA de haut niveau basé sur l'extension *Modular C* [14] de la version C11 du langage de programmation C. *Modular C* met à disposition des numériciens les acquis récents de C, rendant notamment possible la définition des interfaces fonctionnelles génériques sans altérer les performances reconnues à ce langage.

`Arbogast` permet à l'utilisateur d'écrire un code φ indépendamment du contexte numérique, c'est-à-dire sans spécifier ni la précision des calculs ni la nature de ceux-ci (évaluation ou dérivation) par exemple. En fonction du contexte, le code φ est réinterprété comme une fonction C numérique classique ou bien comme un opérateur différentiel appliqué à des polynômes de Taylor. Cette « contextualisation » et cette « réinterprétation » de code font d'`arbogast` un outil de DA atypique, que l'on peut considérer comme étant à mi-chemin entre ceux qui procèdent par surcharge d'opérateurs comme décrit ci-dessus et ceux, dits de transformation de source, qui permettent de dériver les instructions du code φ textuellement pour générer les instructions du code linéarisé φ_1.

La méthode de Héron pour l'approximation de $\sqrt[kappa]{x}$ est choisie pour illustrer cet article. Le code C de référence implantant Héron pour une variable x de type double est présenté dans le Code 1. Il peut être compilé en C et en *Modular C*.

Code 1 – Méthode de Héron pour l'approximation de $\sqrt[kappa]{x}$ implantée en C

```
1 double phi(double x, unsigned kappa) {
2   if (kappa == 1) return x;                          // special case
3   double const chi = 1/x;
4   double rho = (1+x)/kappa;                          // initial estimation
5   for (size_t i = 0; i < imax; ++i) {
6     double rhokappa = powk(rho, kappa-1);            // powk(x, k) = x^k, k integer
7     if (abs2(1-rho*rhokappa*chi) < epsilon) break;
8     rho = ((kappa-1)*rho + x/rhokappa)/kappa; }      // next Heron iteration value
9   return rho; }
```

Comme le montre le Code 2, *Modular C* permet d'introduire de la généralité dans la programmation en recourant à un type générique, ici noté `RC`, correspondant à un des six types flottants. Comme C11 permet l'utilisation d'interfaces génériques, les six fonctions correspondantes sont générées lors de la compilation. Aussi, *Modular C* permet d'utiliser des caractères spéciaux pour une écriture formelle des équations mathématiques.

Code 2 – Méthode générique (réelle ou complexe) augmentée grâce à Modular C, calculant une branche de $\sqrt[\kappa]{x}$

```
RC φ(RC x, unsigned κ) {                  // replace greek names by greek symbols
  if (κ == 1) return x;
  RC const χ = 1/x;
  RC ρ = (1+x)/κ;
  for (size_t i = 0; i < imax; ++i) {
    RC ρκ = powk(ρ, κ-1);                 // powk is a type generic function
    if (abs2(1-ρ*ρκ*χ) < ϵ) break;        // abs2 is a type generic function
    ρ = ((κ-1)*ρ + x/ρκ)/κ; }
  return ρ; }
```

Dans le Code 3 le type générique `TRC` est, selon le contexte, soit un des six types réels ou complexes flottants de C, soit un des six types « Taylor » correspondants.

Comme la surcharge d'opérateurs est absente de C, `arbogast` repose sur une syntaxe à base de fonctions, y compris pour les opérateurs arithmétiques. Leur « contextualisation » s'effectue intégralement lors de la compilation du code, c'est-à-dire sans typage dynamique, pour garantir l'efficacité du code exécutable résultant. Comme les autres bibliothèques de DA, `arbogast` comprend les méthodes permettant de différentier les fonctions élémentaires classiques. Les parenthèses ⟪...⟫ ont étés choisies pour entourer les expressions qui seront transcrites sous forme d'appel à fonctions par Modular C au moment de la compilation. Ceci permet alors la surcharge de ces fonctions par `arbogast`.

Code 3 – Code `arbogast` pour le calcul de $\sqrt[\kappa]{x}$ ou de l'estimation $\sqrt[\kappa]{x_0 + x_1 t + x_2 t^2 + \cdots}$

```
TRC φ(TRC x, unsigned κ) {
  if (κ == 1) return x;
  TRC const χ = ⟪1/x⟫;                  // may be polynomial division
  TRC ρ = ⟪(1+x)/κ⟫;                    // may be division of polynomial by integer
  for (size_t i = 0; i < imax; ++i) {   // for loop uses conventional operations
    TRC ρκ = powk(ρ, κ-1);              // powk now is type generic on TRC
    if (⟪abs2(1-ρ*ρκ*χ) < ϵ⟫) break;    // abs2 returns real type corresponding to TRC
    ρ = ⟪((κ-1)*ρ + x/ρκ)/κ⟫; }
  return ρ; }
```

Les tâches principales de l'utilisateur d'`arbogast` sont d'identifier les variables de type Taylor, de veiller à ce que toutes les interfaces génériques (`powk` et `abs2`, par exemple) puissent considérer des arguments de ce type, et de marquer les expressions qui doivent être transcrites.

5 Conclusions

Partiellement soulignées dans la Section 3, les contributions d'Arbogast dans le domaine de la différentiation automatique sont nombreuses. L'esprit du calcul des dérivations a persisté au cours du temps, guidant les développements théoriques et leur implantation.

L'outil de DA `Arbogast` a été validé par reproduction numérique [8] des résultats obtenus lors du calcul des paramètres de dispersion de haut degré [6] de matériaux diélectriques à l'aide du modèle de Brendel-Bormann reposant sur la fonction de Kramp. Le temps de calcul du code différentié avec *arbogast* est inférieur d'environ 30% à celui observé précédemment.

Arbogast, mathématicien et révolutionnaire alsacien (1859–1803)

Louis François Antoine Arbogast naît en 1759 à Mutzig, bourg de la province d'Alsace situé à vingt kilomètres à l'ouest de Strasbourg. Après des études de droit, il exerce plusieurs années à Colmar, auprès du Conseil souverain d'Alsace. Comme l'atteste sa correspondance, Arbogast s'est également spécialisé en mathématiques.

Arbogast acquiert une renommée internationale à l'attribution des prix de l'Académie de Mantoue pour son *Mémoire sur les principes généraux de la Mécanique* en 1786, et de l'Académie de Saint Petersbourg pour son *Mémoire sur la nature des fonctions arbitraires qui entrent dans les intégrales des équations différentielles partielles* en 1789. La même année, Arbogast présente à l'Académie royale des sciences (Paris) un mémoire sur de nouveaux principes de calcul différentiel indépendants de la théorie des infiniment petits. Entre temps, Arbogast est nommé professeur de mathématiques à l'Ecole royale d'artillerie de Strasbourg et professeur de physique au Collège Royal de la même ville.

La Révolution Française bouleverse tout. Arbogast fait partie des nouveaux notables du Conseil général de Strasbourg, puis est élu en 1791 député du tout nouveau département du Bas-Rhin à l'Assemblée Législative, puis à la Convention nationale en 1792. Arbogast est élu au Comité d'instruction publique, où il y est notamment en charge de rapports et projets de décret sur la composition des livres élémentaires destinés à l'instruction publique (1792), et sur l'uniformité et le système général des poids et mesures (1793).

En 1795, Arbogast prend la direction de l'Ecole centrale de Strasbourg. Il se consacre à la rédaction « du Calcul des Dérivations ». A son décès (1803), il laisse derrière lui une inestimable collection de manuscrits et textes scientifiques, dont des copies des textes de Fermat utilisés pour l'édition des œuvre de ce dernier en 1926.

Références

[1] M. Abramowitz and I. A. Stegun. *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*. Dover Publications, New York, 1964.

[2] L. F. A. Arbogast. *Du calcul des dérivations*. Imprimerie de Levrault frères, Strasbourg, An VIII (1800).

[3] E. T. Bell. Exponential polynomials. *Ann. Math.*, 35 :258–277, 1934.

[4] B. Brisson. Sur l'intégration des équations différentielles partielles. *Journal de l'Ecole Polytechnique*, 7 :191–261, 1808.

[5] I. Charpentier. On higher-order differentiation in nonlinear mechanics. *Optim. Method. Softw.*, 27 :221–232, 2012.

[6] I. Charpentier. Optimized higher-order automatic differentiation for the Faddeeva function. *Comput. Phys. Commun.*, 2016. to appear.

[7] I. Charpentier, C. Dal Cappello, and J. Utke. Efficient higher-order derivatives of the hypergeometric function. In C. H. Bischof et al., editors, *Advances in Automatic Differentiation*, pages 127–137. Springer, 2008.

[8] I. Charpentier and J. Gustedt. Arbogast : Higher order AD for special functions with Modular C. Research Report 8907, Inria Nancy - Grand Est (Villers-lès-Nancy, France), Apr. 2016.

[9] B. Christianson. Reverse accumulation and attractive fixed points. *Optim. Methods Softw.*, 3 :311–326, 1994.

[10] A. D. D. Craik. Prehistory of Faà Di Bruno's Formula. *Amer. Math. Monthly*, 112 :119–130, 2005.

[11] F. Faà de Bruno. Note sur une nouvelle formule de calcul differentiel. *Quart. J. Pure Appl. Math.*, 1 :359–360, 1857.

[12] H. Flanders. Automatic differentiation : Origin and references, 2002.

[13] A. Griewank and A. Walther, editors. *Evaluating Derivatives : Principles and Techniques of Algorithmic Differentiation*. SIAM, Philadelphia, second edition, 2008.

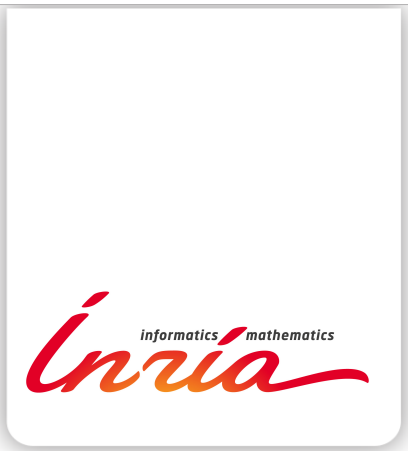
[14] J. Gustedt. Modular C. Research Report RR-8751, INRIA, June 2015.

[15] W. P. Johnson. The curious history of Faa di Bruno's formula. *Am. Math. Monthly*, 109 :217–234, 2002.

[16] C. Kramp. *Eléments d'arithmétique universelle*. Imprimerie de Th. F. Thiriart, Cologne, 1808.

[17] J.-P. Lubet and J.-P. Friedelmeyer. *L'analyse algébrique, un épisode clé de l'histoire des mathématiques*. Collection : IREM - Histoire des mathématiques. Ellipses, 2014.

[18] O. Terquem. *Supplément des Nouvelles Annales de Mathematiques*, volume 14. 1855. p. 83–93.



RESEARCH CENTRE
NANCY – GRAND EST

615 rue du Jardin Botanique
CS20101
54603 Villers-lès-Nancy Cedex

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr

ISSN 0249-6399